Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Инженерно-технологическая академия ЮФУ

Институт компьютерных технологий и информационной безопасности

**Кафедра информационно-аналитических систем безопасности**

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

по курсу: «Алгоритмизация и программирование»

Выполнил

студент группы КТсо1-5

Гладков Р.Е

№ зачетки КТ-23-0340

Таганрог, 2023

# Техническое задание

## Задача задания

Задание состоит в разработке программы, которая считывает  настроечные параметры и формирует отчет по имеющимся данным в файлах. Отчет записывается в текстовый файл. Число записей в каждом из описанных выше файлов произвольно.

## Задание

Вариант № 4

Отчет: Получить перечень услуг,  предоставленных в текущем месяце, с суммарным значением того сколько было потрачено клиентом(клиентами) в заданном диапазоне значений, в диапазон должно попадать значение, того сколько потратил один пользователь на сервис.

Параметры: Минимальное значение и максимальное значение диапазона.

# Выполнение работы

Интегрированная среда разработки Visual Studio 2022 включает в себя текстовый редактор, компилятор, отладчик, среду для обмена проектами. Позволяет создавать проекты, веб-сайты, файлы исходного кода, разрабатывать программное обеспечение на языках программирования C, C++, C#, F#

**Исходный код программы**

#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

#include <string>

#include <map>

#include <algorithm>

#include <ctime> // Для работы с временем

#include <chrono> // Для преобразования даты-времени

#include <iomanip> // Для форматирования вывода даты-времени

#pragma warning(disable : 4996)

using namespace std;

std::string read\_ini\_file(const std::string& filename, const std::string& key) {

std::ifstream file(filename);

std::string line;

while (std::getline(file, line)) {

if (line.find(key) != string::npos) {

size\_t start = line.find('=') + 1;

size\_t end = line.length();

return line.substr(start, end - start);

}

}

return "";

}

struct Client {

std::string fullName;

std::string phoneNumber;

std::string startDate;

std::string endDate;

int debt;

int creditLimit;

};

struct Service {

std::string name;

int code;

double rate;

std::string interval;

};

struct Usage {

std::string phoneNumber;

int serviceCode;

std::string dateTime;

int durationSeconds;

};

int SecondsInMonth() {

//Cчитаем сколько секунд в месяце

int month;

int yearday;

std::time\_t now = std::time(NULL);

std::tm\* localTime = std::localtime(&now);

month = localTime->tm\_mon + 1;

yearday = localTime->tm\_yday + 1;

if (month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12) {

return (2678400);

}

else if (month == 4 || month == 6 || month == 9 || month == 11) {

return (2592000);

}

else {

if (yearday == 365) {

return (2419200);

}

else {

return (2505600);

}

}

}

bool IsDateInCurrentMonth(const std::string& dateTime) {

// Получаем текущую дату и время

std::time\_t now = std::time(NULL);

std::tm\* localTime = std::localtime(&now);

// Парсим dateTime

std::istringstream ss(dateTime);

int year, month, day, hour, minute, second;

char delimiter;

ss >> year >> delimiter >> month >> delimiter >> day >> hour >> delimiter >> minute >> delimiter >> second;

// Если месяцы совпадают, возвращаем true, иначе false

return (month == localTime->tm\_mon + 1); // tm\_mon возвращает месяц от 0 до 11

}

int main() {

std::vector<Client> clients;

std::vector<Service> services;

std::vector<Usage> usages;

std::vector<int> range;

std::map<int, double> serviceRates;

//setlocale(LC\_ALL, "Russian");

// Чтение информации о клиентах

std::ifstream clientsFile("Clients.txt");

if (clientsFile.is\_open()) {

std::string line;

while (std::getline(clientsFile, line)) {

std::istringstream ss(line);

std::string token;

Client client;

for (int i = 0; std::getline(ss, token, ','); ++i) {

switch (i) {

case 0: client.fullName = token; break;

case 1: client.phoneNumber = stod(token); break;

case 2: client.startDate = token; break;

case 3: client.endDate = token; break;

case 4: client.debt = std::stoi(token); break;

case 5: client.creditLimit = std::stoi(token); break;

}

}

clients.push\_back(client);

}

clientsFile.close();

}

else {

std::cerr << "Error: Unable to open clients file." << std::endl;

return 1;

}

// Чтение информации об услугах

std::ifstream servicesFile("Services.txt");

if (servicesFile.is\_open()) {

std::string line;

while (std::getline(servicesFile, line)) {

std::istringstream ss(line);

std::string token;

Service service;

for (int i = 0; std::getline(ss, token, ','); ++i) {

switch (i) {

case 0: service.name = token; break;

case 1: service.code = std::stoi(token); break;

case 2: service.rate = std::stod(token); break; // Используем stod для считывания значения с плавающей точкой

case 3: service.interval = token; break;

}

}

services.push\_back(service);

}

servicesFile.close();

}

else {

std::cerr << "Error: Unable to open services file." << std::endl;

return 1;

}

// Чтение информации об использовании услуг

std::ifstream usageFile("Usage.txt");

if (usageFile.is\_open()) {

std::string line;

while (std::getline(usageFile, line)) {

std::istringstream ss(line);

std::string token;

Usage usage;

for (int i = 0; std::getline(ss, token, ','); ++i) {

switch (i) {

case 0: usage.phoneNumber = stod(token); break;

case 1: usage.serviceCode = std::stoi(token); break;

case 2: usage.dateTime = token; break;

case 3: usage.durationSeconds = std::stoi(token); break;

}

}

if (IsDateInCurrentMonth(usage.dateTime)) { // Проверяем, что дата в текущем месяце

usages.push\_back(usage);

}

}

usageFile.close();

}

else {

std::cerr << "Error: Unable to open usage file." << std::endl;

return 1;

}

// Чтение диапазона параметров.

std::string filename = "Param.ini";

std::string key1 = "key1";

std::string key2 = "key2";

std::string value1 = read\_ini\_file(filename, key1);

std::string value2 = read\_ini\_file(filename, key2);

// Создание словаря тарифов услуг

for (const auto& service : services) {

double modifiedRate = service.rate; // Значение по умолчанию

// Изменяем тариф в зависимости от интервала

if (service.interval == " мин") {

modifiedRate /= 60;

}

else if (service.interval == " час") {

modifiedRate /= 3600;

}

else if (service.interval == " сутки") {

modifiedRate /= 86400;

}

else if (service.interval == " месяц") {

modifiedRate /= SecondsInMonth();

}

else {

modifiedRate /= 1; // Если интервал не определен, используем значение по умолчанию

}

// Внесем скорректированный тариф в словарь serviceRates

serviceRates[service.code] = modifiedRate;

}

// Вычисление потраченной суммы.

std::ofstream reportFile("Report.txt");

bool flag = false;

std::map<int, double> totalCostPerService;

for (const auto& usage : usages) {

for (const auto& client : clients) {

if (client.phoneNumber == usage.phoneNumber) {

auto it = serviceRates.find(usage.serviceCode);

if (it != serviceRates.end()) {

double cost = it->second \* usage.durationSeconds;

if (cost >= stod(value1) && cost <= stod(value2)) {

totalCostPerService[usage.serviceCode] += cost;

}

}

}

}

}

for (const auto& pair : totalCostPerService) {

// Ищем название сервиса по его коду

std::string serviceName;

for (const auto& service : services) {

if (service.code == pair.first) {

serviceName = service.name;

break;

}

}

flag = true;

reportFile << "Сервис: " << serviceName << " (" << pair.first << "), Сумма: " << pair.second << " рублей" << std::endl;

}

if (flag) {

cout << "Report created successfully!" << endl;

}

else {

cout << "The report was not created!" << endl;

}

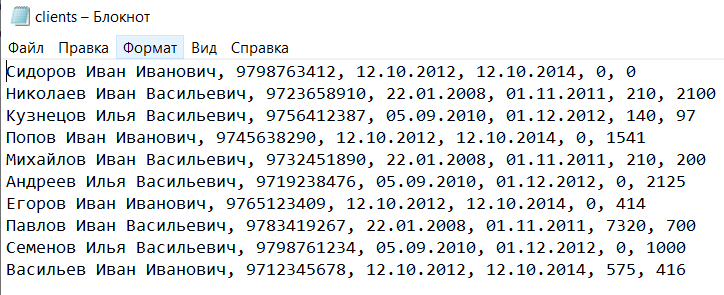
reportFile.close();

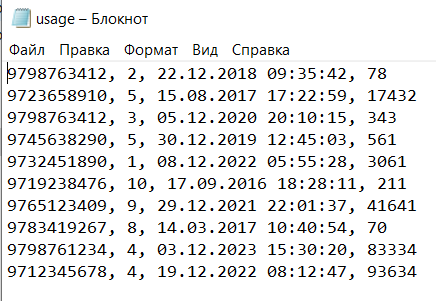
return 0;

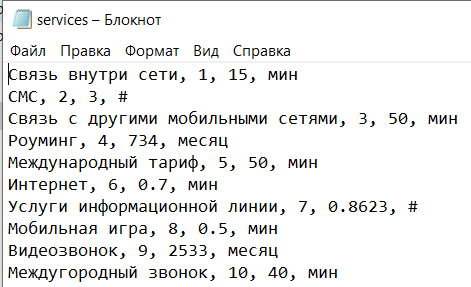
}

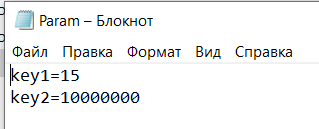
# Пример работы программы

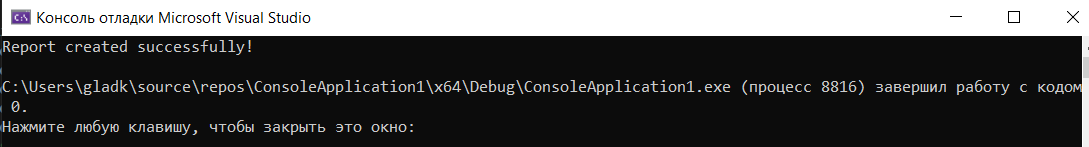
На изображениях показаны изначальные данные и скриншоты работы программы в файле.

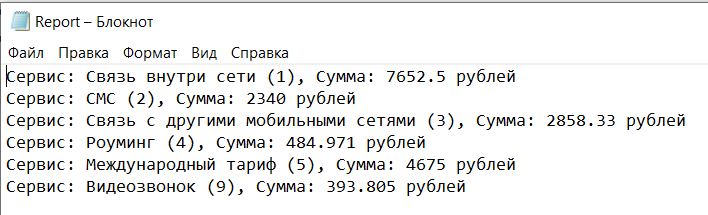












**Вывод**

Проделав индивидуальное задание, я научился работать с несколькими файлами, обрабатывать их, научился работать со словарями, узнал как использовать библиотеки получения и обработки времени, закрепил знания о структурах, циклах, векторах.